project_2

krishna sai surendra babu kalluri

27/07/2020

Bias correction of numerical prediction model temperature forecast Data Set

Introduction:

The purpose of this Dataset is used for predicting next day minimum and maximum air temperatures.But a Dataset is not all about predicting only future.Instead ,we need to predict how well our model fit into the data .what are independent variables and dependent variables and which variables has influence on our Dataset and which variable is useful for predicting the y.

Expectation of following day minimum and maxiumum air temperatures is fundamental in tropical nations like India where essential occupation is Agriculture.There are approaches to anticipate the temperatures typically however the accuracy rate is low for them so it might get off-base sometimes.But foreseeing temperatures by taking a Dataset ,reading the previous data and analysing it can improve the accuracy rate which makes me to pick this dataset.

This dataset is available in https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast

This dataset is operated by Korea Meteorological Administration over Seoul, South Korea.It has 7750 observations and number of attributes is 25.

Nature of Dataset :

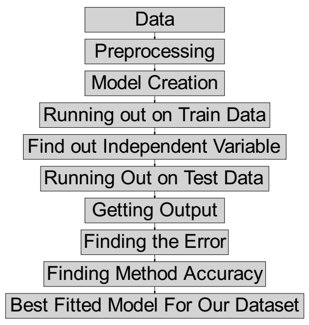
- Data Set Characteristics : Multivariate
- Attribute Characteristics: Real
- Area : Physical
- Number of Instances : 7750
- Number of Attributes : 25
- Associated Tasks : Regression

Attribute	Information
station	used weather station number: 1 to 25
Date-present day	yyyy-mm-dd
Present_Tmax	Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6
Present_Tmin	Minimum air temperature between 0 and 21 h on the present day (Â°C): 11.3 to 29.9

Attribute	Information
LDAPS_RHmin	LDAPS model forecast of next-day minimum relative humidity (%): 19.8 to 98.5
LDAPS_RHmax	LDAPS model forecast of next-day maximum relative humidity (%): 58.9 to 100
LDAPS_Tmax_lapse	LDAPS model forecast of next-day maximum air temperature applied lapse rate (Â°C): 17.6 to 38.5
LDAPS_Tmin_lapse	LDAPS model forecast of next-day minimum air temperature applied lapse rate (Â°C): 14.3 to 29.6
LDAPS_WS	LDAPS model forecast of next-day average wind speed (m/s): 2.9 to 21.9
LDAPS_LH	LDAPS model forecast of next-day average latent heat flux (W/m2): -13.6 to 213.4
LDAPS_CC1	LDAPS model forecast of next-day 1st 6-hour split average cloud cover (0-5 h) (%): 0 to 0.97
LDAPS_CC2	LDAPS model forecast of next-day 2nd 6-hour split average cloud cover (6-11 h) (%): 0 to 0.97
LDAPS_CC3	LDAPS model forecast of next-day 3rd 6-hour split average cloud cover (12-17 h) (%): 0 to 0.98
LDAPS_CC4	LDAPS model forecast of next-day 4th 6-hour split average cloud cover (18-23 h) (%): 0 to 0.97
LDAPS_PPT1	LDAPS model forecast of next-day 1st 6-hour split average precipitation (0-5 h) (%): 0 to 23.7
LDAPS_PPT2	LDAPS model forecast of next-day 2nd 6-hour split average precipitation (6-11 h) (%): 0 to 21.6
LDAPS_PPT3	LDAPS model forecast of next-day 3rd 6-hour split average precipitation (12-17 h) (%): 0 to 15.8
LDAPS_PPT4	LDAPS model forecast of next-day 4th 6-hour split average precipitation (18-23 h) (%): 0 to 16.7
lat	Latitude (Â°)
lon	Longitude (Â°)
DEM	Elevation (m)
Slope	Slope (Â°)
Solar radiation	Daily incoming solar radiation (wh/m2): 4329.5 to 5992.9
Next_Tmax	The next-day maximum air temperature (Â°C): 17.4 to 38.9
Next_Tmin	The next-day minimum air temperature (Â°C): 11.3 to 29.8

Overview of Process:

```
knitr::opts_chunk$set(echo = TRUE)
library(DiagrammeR)
grViz("digraph flowchart {
      #graph[fontsize=10]
      # node definitions with substituted label text

      node [fontsize = 100,fontname = Helvetica, shape = rectangle,style=filled,width=10]
      tab1 [label = '@@1']
      tab2 [label = '@@2']
      tab3 [label = '@@3']
      tab4 [label = '@@4']
      tab5 [label = '@@5']
      tab6 [label = '@@6']
      tab7 [label = '@@7']
      tab8 [label = '@@8']
      tab9 [label = '@@9']
      tab10 [label = '@@10']



      # edge definitions with the node IDs
      tab1 -> tab2-> tab3 -> tab4 -> tab5 ->tab6->tab7 ->tab8 ->tab9 ->tab10 ; }

      [1]: 'Data'
      [2]: 'Preprocessing'
      [3]: 'Model Creation'
      [4]: 'Running out on Train Data'
      [5]:'Find out Independent Variable'
      [6]:'Running Out on Test Data'
      [7]:'Getting Output'
      [8]:'Finding the Error'
      [9]:'Finding Method Accuracy'
      [10]:'Best Fitted Model For Our Dataset'

      ")
```



Implementation :

Reading Data :

```
knitr::opts_chunk$set(echo = TRUE)
data <- read.csv("C:\\Users\\kallu\\Desktop\\project2\\Bias_correction_ucl.csv")
```

```
knitr::opts_chunk$set(echo = TRUE)

set.seed(100)
trainingRowIndex <- sample(1:nrow(data), 0.8*nrow(data))  # row indices for training data
trainingData <- data[trainingRowIndex, ]  # model training data
testData  <- data[-trainingRowIndex, ]
```

During Predicting,First we have to Import and read the data.Presently, splitting the data into training data and testing data .Finding out which variables needs to take to foresee by plotting.

Methods :

This Dataset does not require any preprocessing and it is a regression model.So,I am using three methods for the dataset in order to analyse the data how well it fit into the data and predicting which has low error rate.

- GAM
- Decision Trees
- SVM regression
- PCA

GAM:

General additive models will provide a general framework for extending a standard linear model by permitting non linear elements of each of the variables by maintaining additivity. Much the same as direct models, GAMs additivity can be applied with both quantitative and qualitative responses.

```
library(modelr)
library(gam)
require(ISLR)
knitr::opts_chunk$set(echo = TRUE)
#Tmax
gam1<-gam(Next_Tmax~Present_Tmax+LDAPS_RHmin+s(LDAPS_RHmax,df=6)+LDAPS_Tmax_lapse+LDAPS_Tmin_lapse+lat+
summary(gam1)

testData=na.omit(testData)
model1=predict(gam1,newdata = testData)
model1 = as.numeric(model1)

#Tmin
gam2<-gam(Next_Tmin~Present_Tmax+ s(LDAPS_RHmin,df=6)+s(LDAPS_RHmax,df=6)+LDAPS_Tmax_lapse+LDAPS_Tmin_la
gam3<-gam(Next_Tmin~Present_Tmax+ LDAPS_RHmin+s(LDAPS_RHmax,df=6)+LDAPS_Tmax_lapse+LDAPS_Tmin_lapse+lat
```

```r
summary(gam2)
anova(gam2,gam3)
model2=predict(gam3,newdata = testData)
model2=as.numeric(model2)
```

After taking the variables ,we created a GAM model with our variables on training dataset in order to predict the Next_Tmax . Similarly,we created a model for Next_Tmin .Here we take the two models and find the best model to predict the Tmin.Now We started predicting the models on test data and will find out the error rate.

Decision Trees :

In general, Tree-based methods are simple and are very useful for interpretation.Even if it is a good method but it is not competitive with the best supervised learning methods.This type of methods involves producing multiple trees which are combined to form a single prediction .By combining large number of trees can improve accuracy prediction.

```r
library(tree)
library(rpart)
library(ggplot2)
library(rpart.plot)
library(Metrics)
knitr::opts_chunk$set(echo = TRUE)
#Tmax
tree.bias=tree(Next_Tmax ~ .,data=trainingData)
summary(tree.bias)
testData=na.omit(testData)
y1 = predict (tree.bias ,newdata=testData)
y1 = as.numeric(y1)


#Tmin
tree.bias2=tree(Next_Tmin ~ .,data=trainingData)
summary(tree.bias2)
testData=na.omit(testData)
y2 = predict (tree.bias2 ,newdata=testData)
y2 = as.numeric(y2)
```

Here, we created a model using tree which gives us the variables used for tree construction.With this model I am predicting the next tmax and tmin using predict function on test data set.

SVM regression :

It is a statistical method that finds the relationship between two continuous variable.The idea is same as support vector classification and allows the presence of non linearity in a dataset and gives a good prediction model.

```r
library(e1071)
library(caret)
library(magrittr)
library(dplyr)
library(knitr)
library(kableExtra)
library(dplyr)
library(ggplot2)
knitr::opts_chunk$set(echo = TRUE)

#Tmax
model_reg = svm(Next_Tmax~., data=trainingData)
print(model_reg)
testData=na.omit(testData)
pred = predict(model_reg, newdata=testData)
pred= as.numeric(pred)
x = 1:length(testData$Next_Tmax)
#Tmin
model_reg2 = svm(Next_Tmin~., data=trainingData)
print(model_reg2)
testData=na.omit(testData)
pred1 = predict(model_reg2, newdata=testData)
pred1= as.numeric(pred1)
x = 1:length(testData$Next_Tmin)
```

Here , we developed a model using svm on training data for Tmax and Tmin. we have some non NA values in my dataset so I removed them using na.omit function .After that I started predicting my test data and getting error rate.

PCA:

Principal component Analysis,is a method of summarizing the larger dataset into smaller number of representative variables which explains the most variability in the original dataset.It is a unsupervised method and powerful in dealing with multicollinearity and variables.

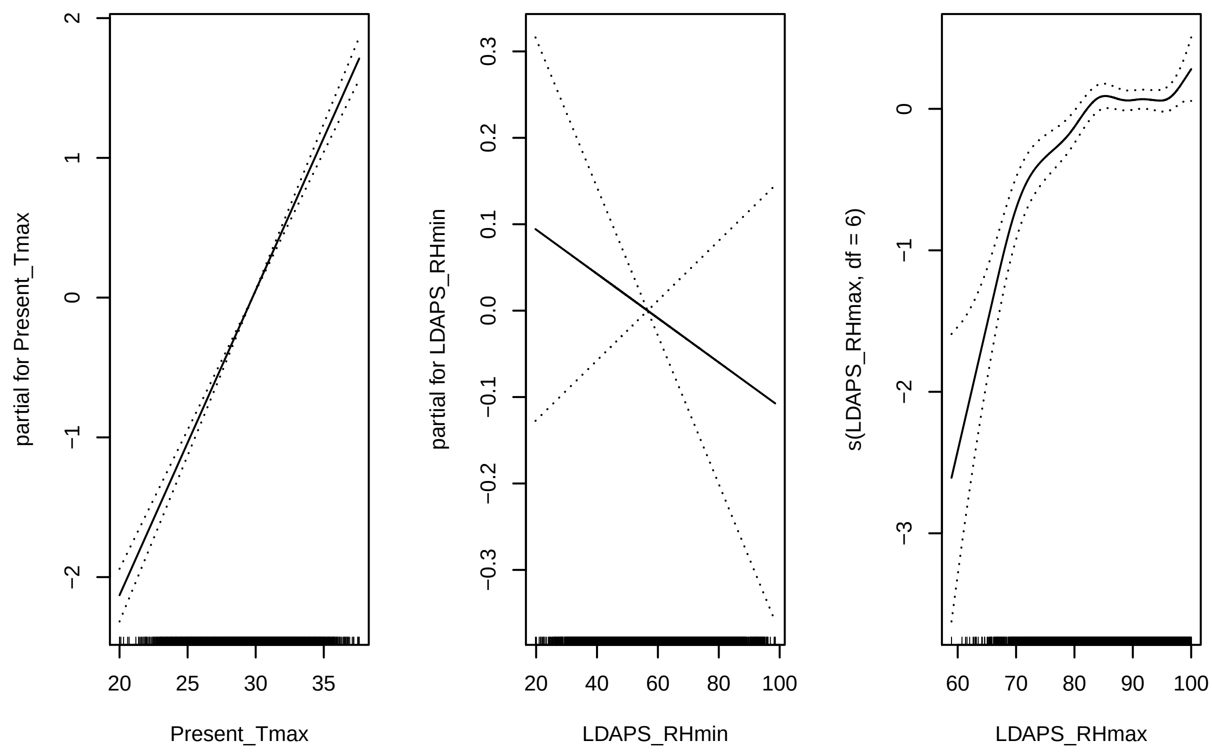
```r
knitr::opts_chunk$set(echo = TRUE)
df <-as.data.frame(data)
df[is.na(df)] <-0
df$Date <- NULL
pcaT<- prcomp(scale(df[,-24]))
TScores <- pcaT$x
#Tmax
modTmax <- lm(df$Next_Tmax ~ TScores[,1:3])
summary(modTmax)

#Tmin
modTmin <- lm(df$Next_Tmin ~ TScores[,1:3])
summary(modTmin)
```

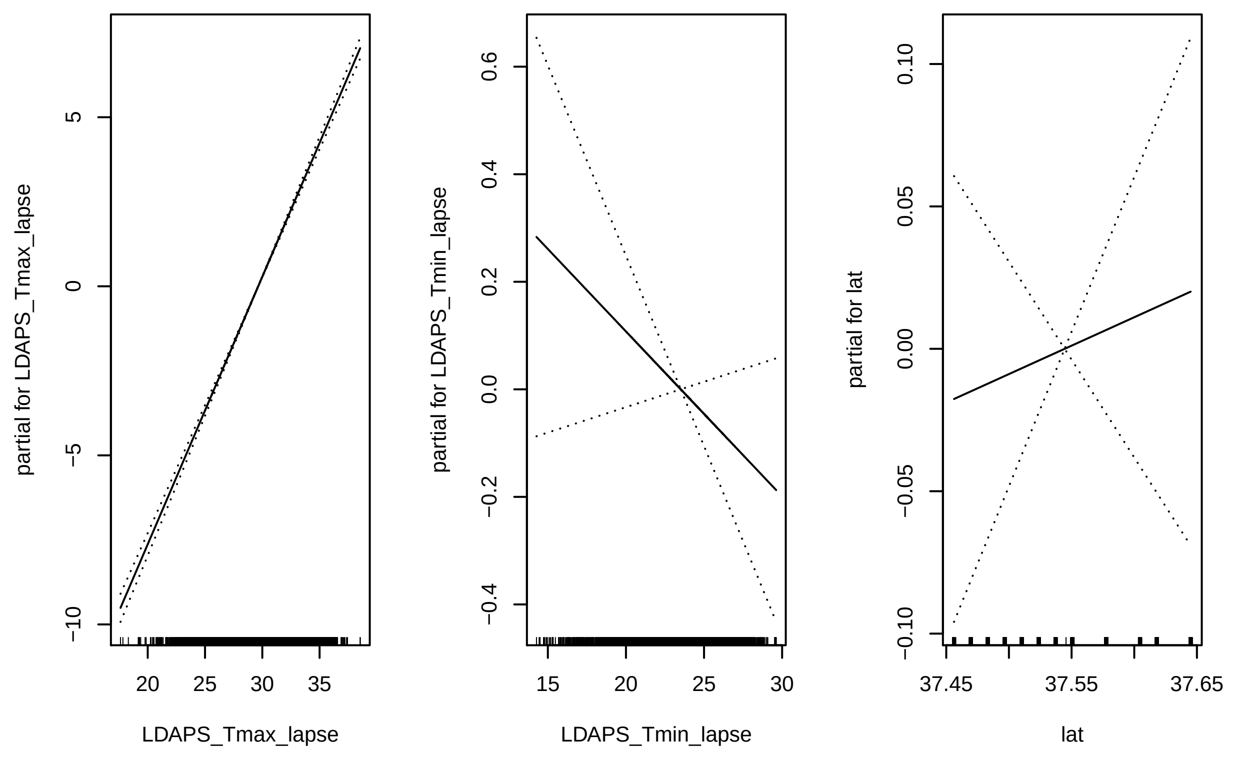
Here,First we removed the Na values fro the dataset and performed the pca method ,then after that we created a linear method for predicting the next tmax and tmin, Calculated the variance and standard deviation and plotted the graphs.

Plots :

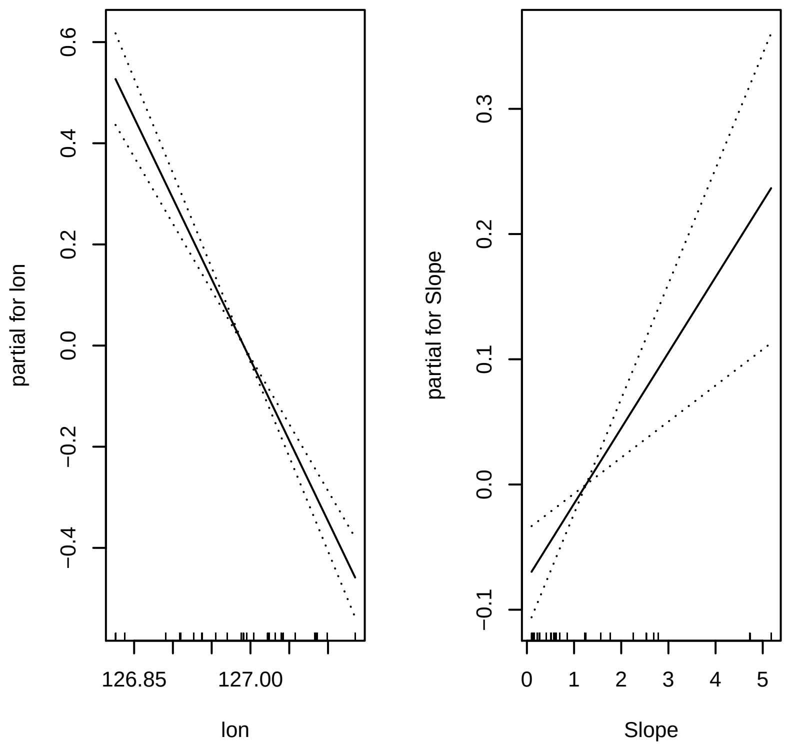
GAM :

```
knitr::opts_chunk$set(echo = TRUE)
#Tmax
par(mfrow=c(1,3))
plot(gam1,se = TRUE)
```

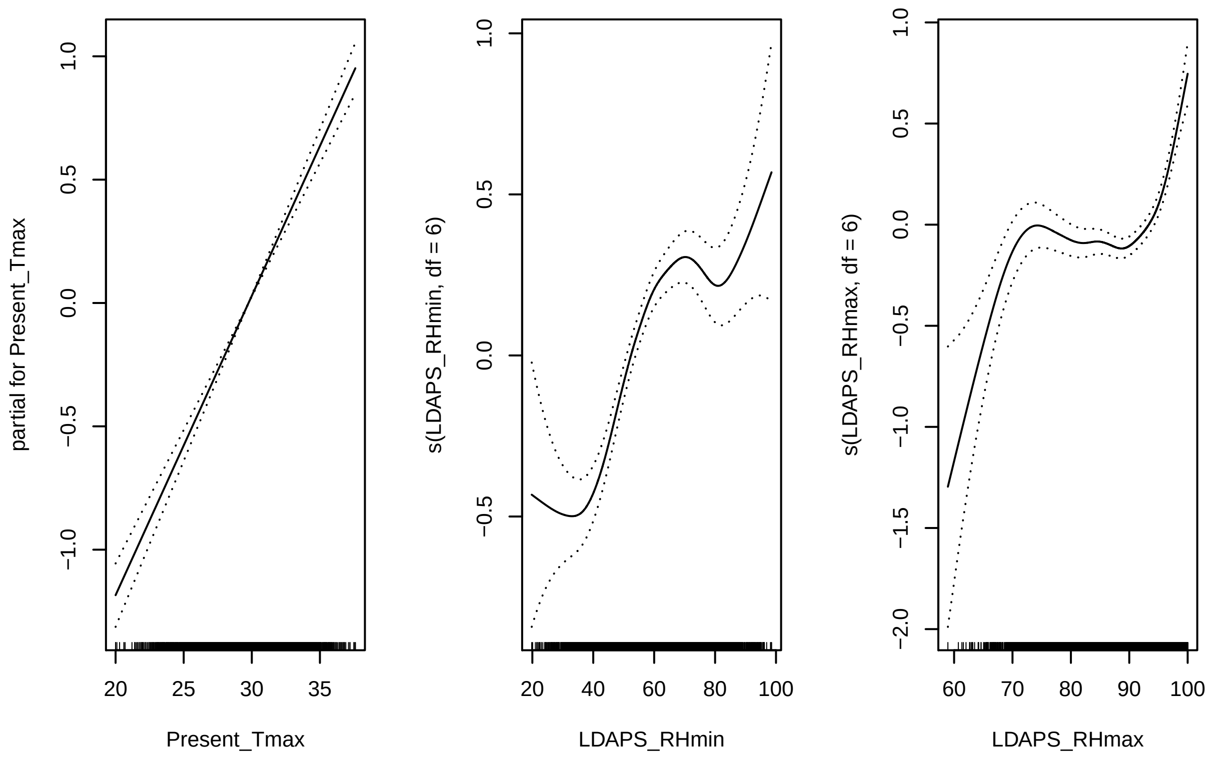


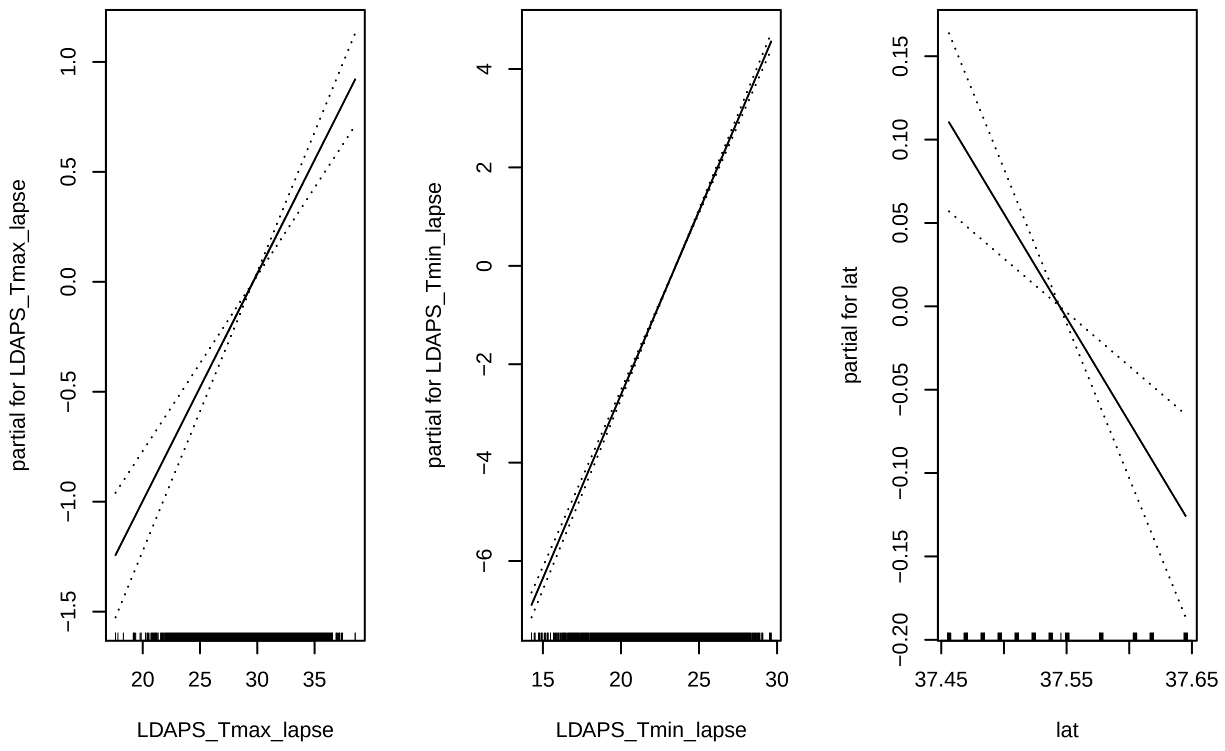


```
#Tmin
par(mfrow=c(1,3))
```

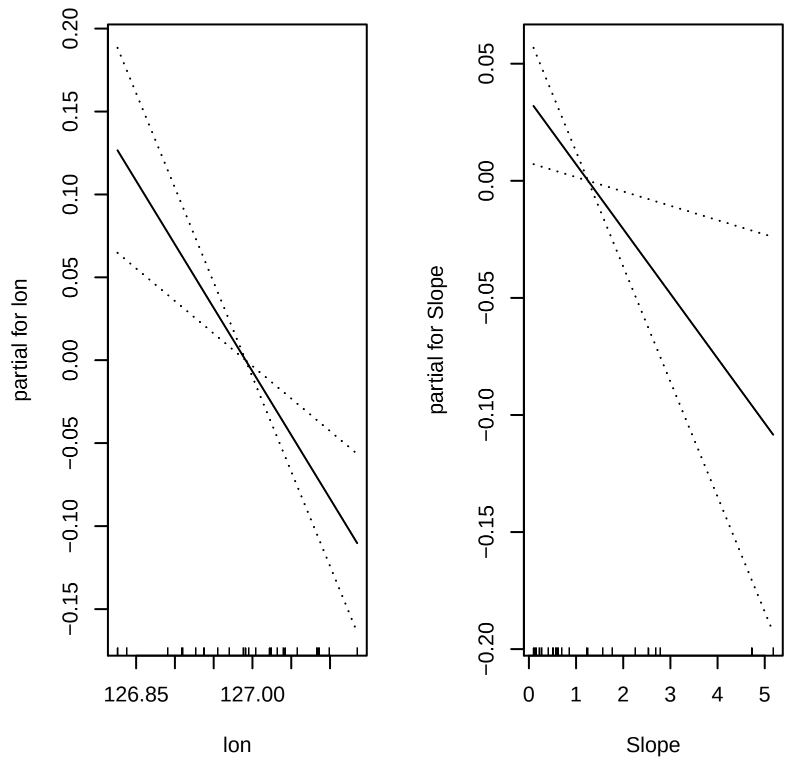


```r
plot(gam2,se = TRUE)
```


partial for Present_Tmax
Present_Tmax
s(LDAPS_RHmin, df = 6)
LDAPS_RHmin
s(LDAPS_RHmax, df = 6)
LDAPS_RHmax

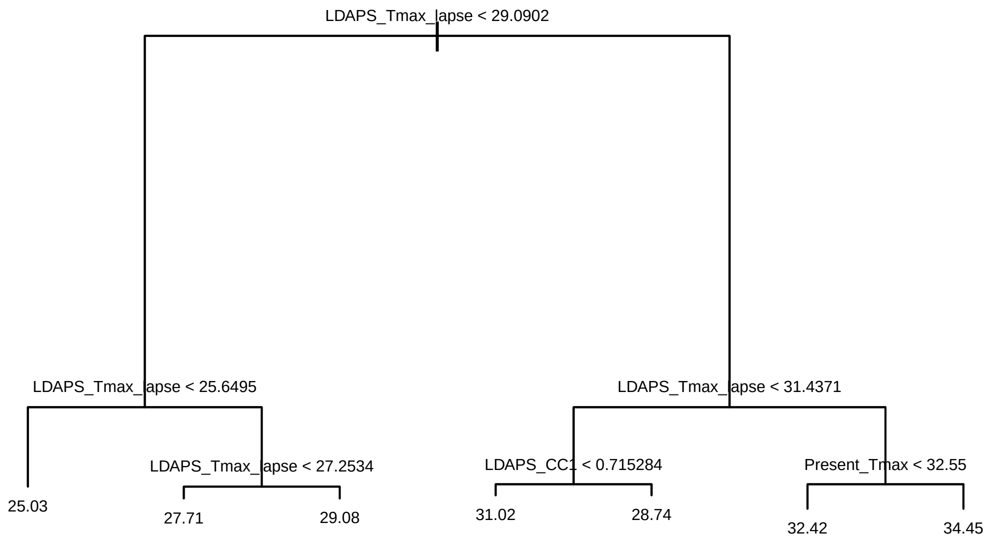


partial for LDAPS_Tmax_lapse
−1.5
−1.0
−0.5
0.0
0.5
1.0
20
25
30
35
LDAPS_Tmax_lapse
partial for LDAPS_Tmin_lapse
−6
−4
−2
0
2
4
15
20
25
30
LDAPS_Tmin_lapse
partial for lat
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
37.45
37.55
37.65
lat

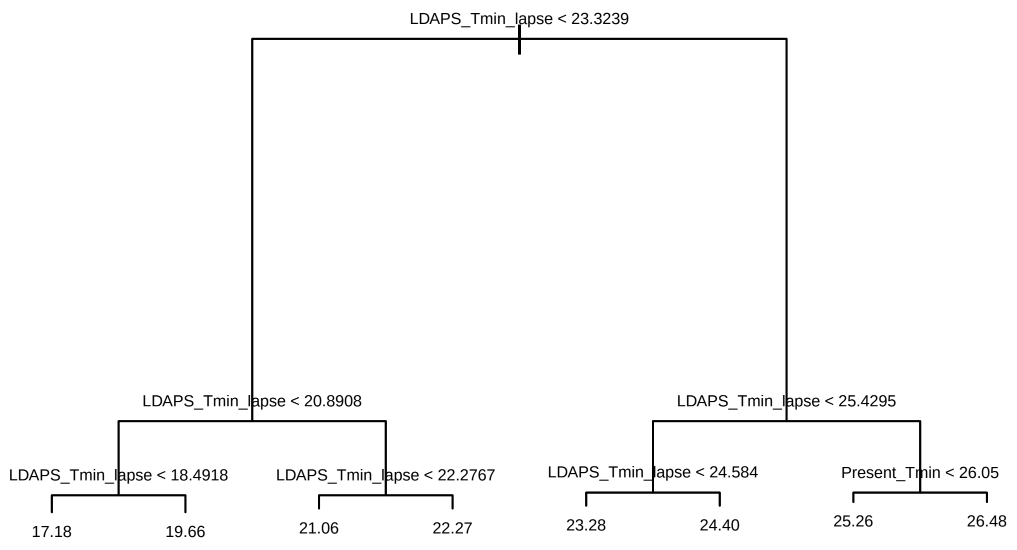


Decision Tree:

```r
knitr::opts_chunk$set(echo = TRUE)
#Tmax
plot(tree.bias)
text(tree.bias, cex=0.5)
```

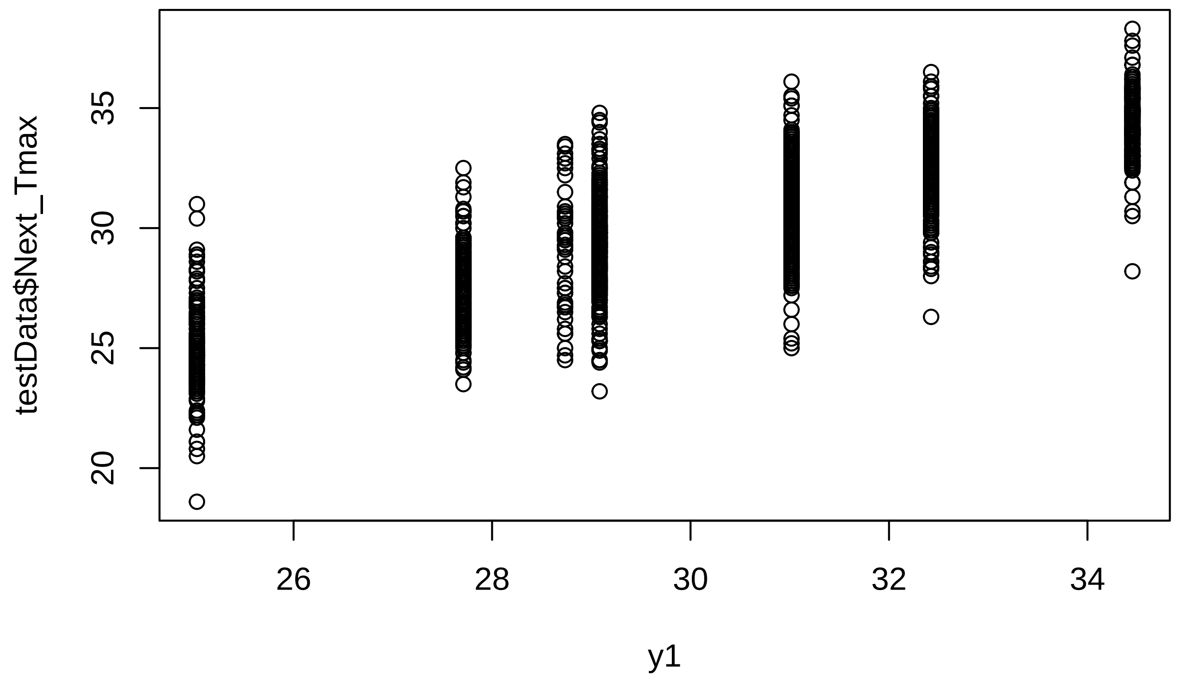

LDAPS_Tmax_lapse < 29.0902
LDAPS_Tmax_lapse < 25.6495
LDAPS_Tmax_lapse < 31.4371
LDAPS_Tmax_lapse < 27.2534
LDAPS_CC1 < 0.715284
Present_Tmax < 32.55
25.03
27.71
29.08
31.02
28.74
32.42
34.45

```r
#Tmin
plot(tree.bias2)
text(tree.bias2, cex=0.5)
```

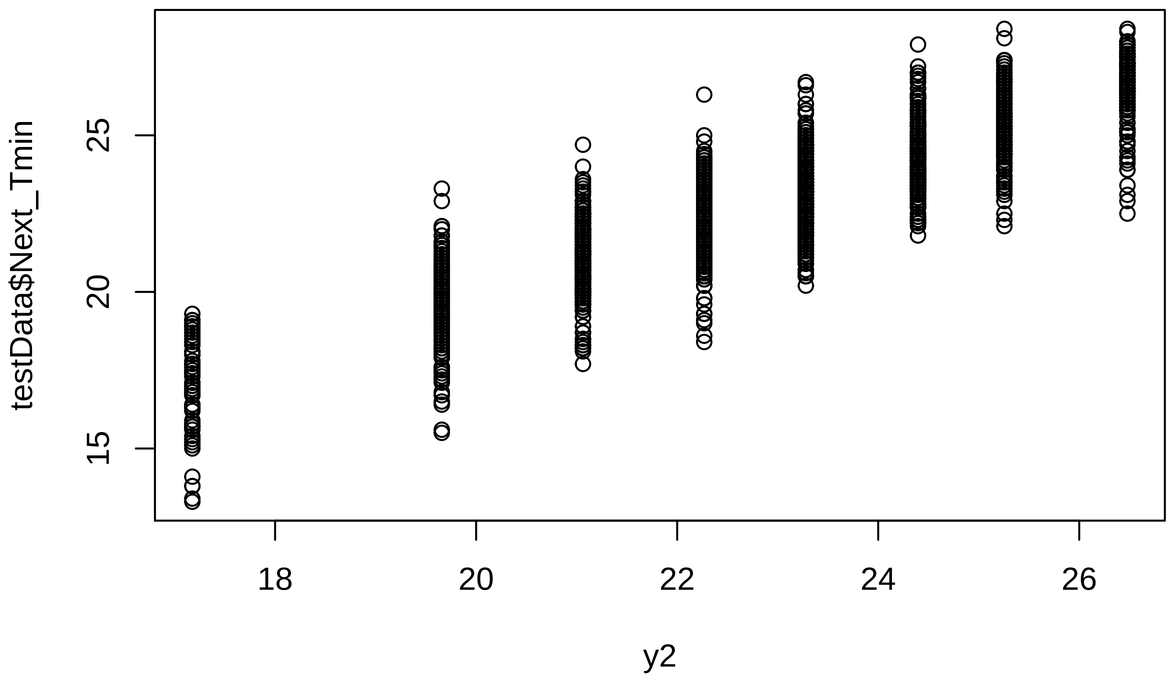

LDAPS_Tmin_lapse < 23.3239
LDAPS_Tmin_lapse < 20.8908
LDAPS_Tmin_lapse < 25.4295
LDAPS_Tmin_lapse < 18.4918
LDAPS_Tmin_lapse < 22.2767
LDAPS_Tmin_lapse < 24.584
Present_Tmin < 26.05
17.18
19.66
21.06
22.27
23.28
24.40
25.26
26.48

#### Prediction plots:

```
knitr::opts_chunk$set(echo = TRUE)
#Tmax
#predicting plots
plot(y1 ,testData$Next_Tmax)
```

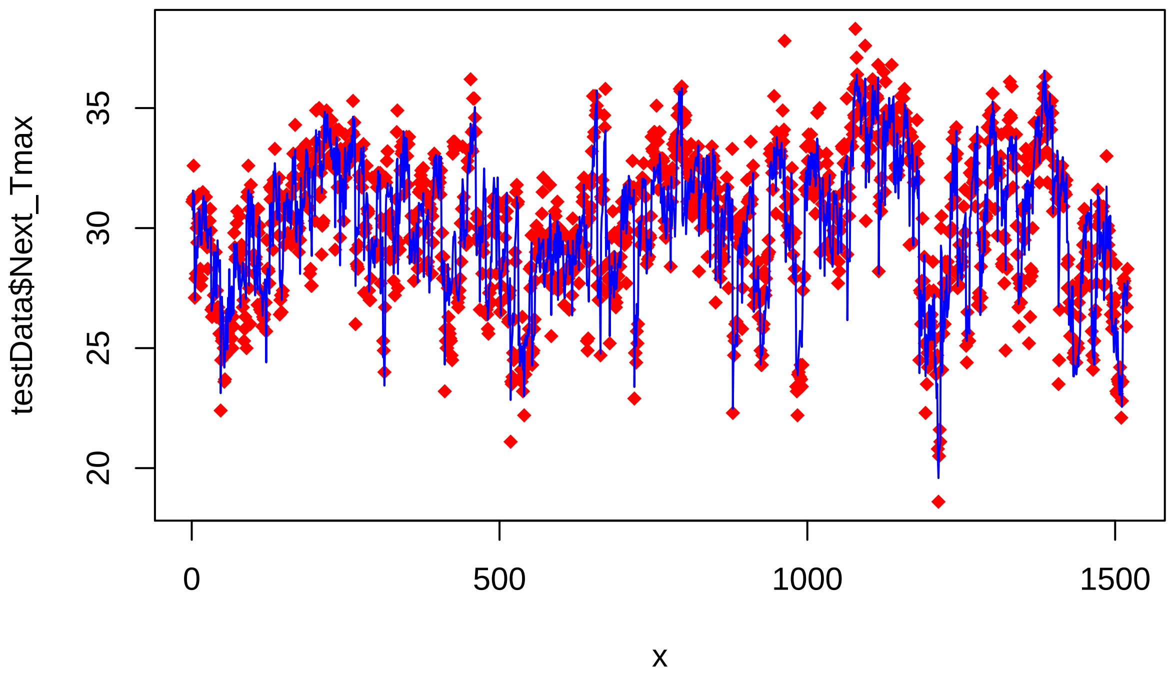


```
#Tmin
#predicting plots
plot(y2 ,testData$Next_Tmin)
```

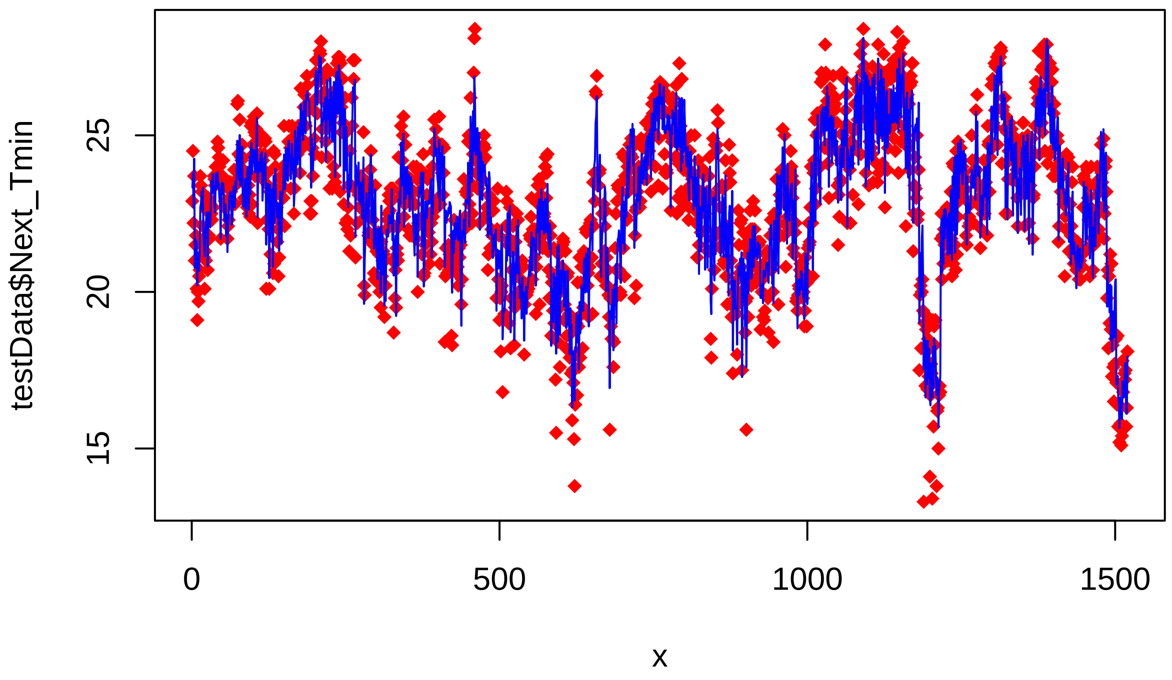


SVM :

```
knitr::opts_chunk$set(echo = TRUE)
#Tmax
plot(x, testData$Next_Tmax, pch=18, col="red")
lines(x, pred, lwd="1", col="blue")
```

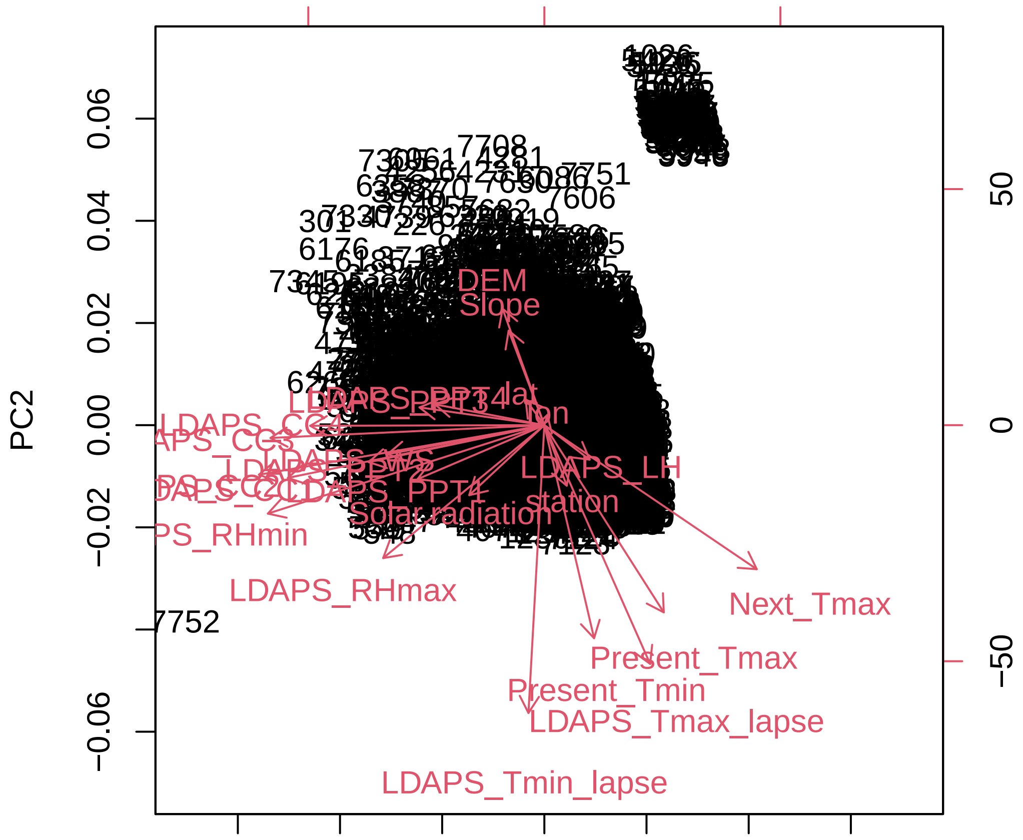


```
#Tmin
plot(x, testData$Next_Tmin, pch=18, col="red")
lines(x, pred1, lwd="1", col="blue")
```

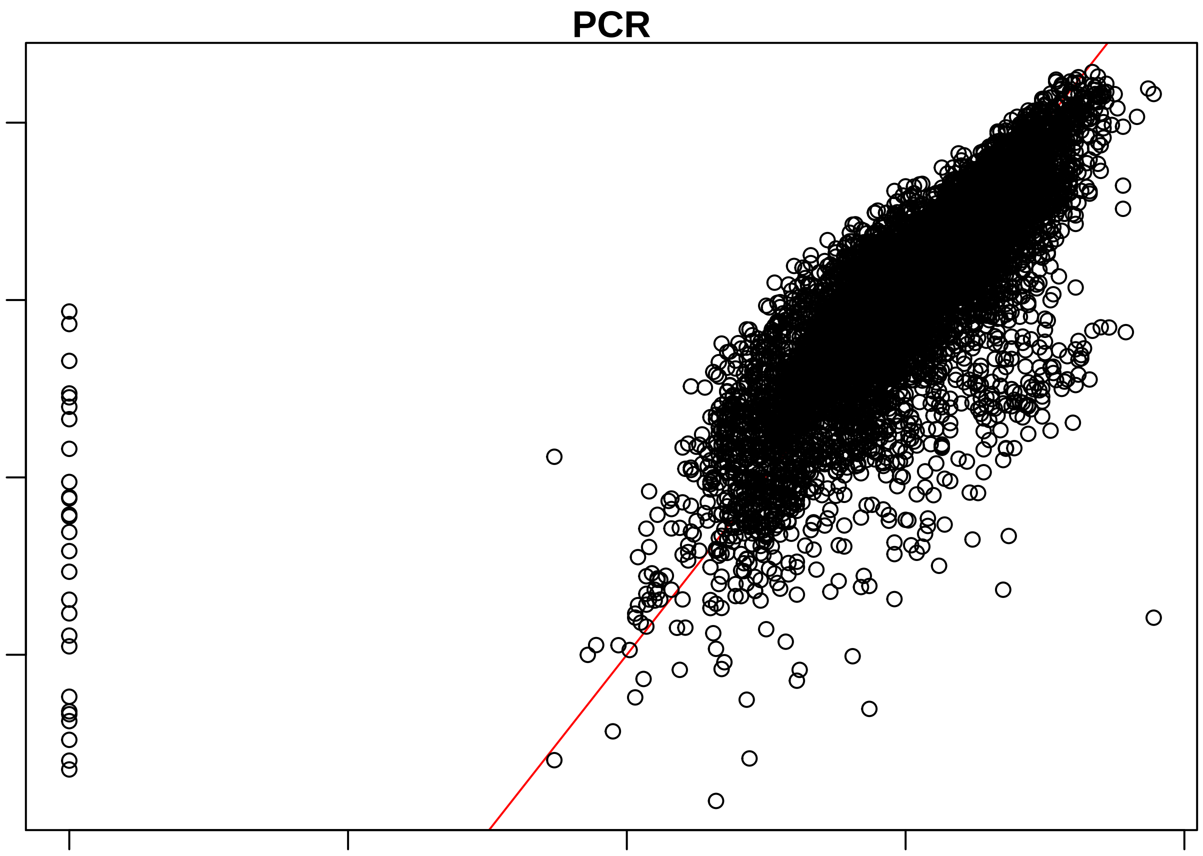


PCA:

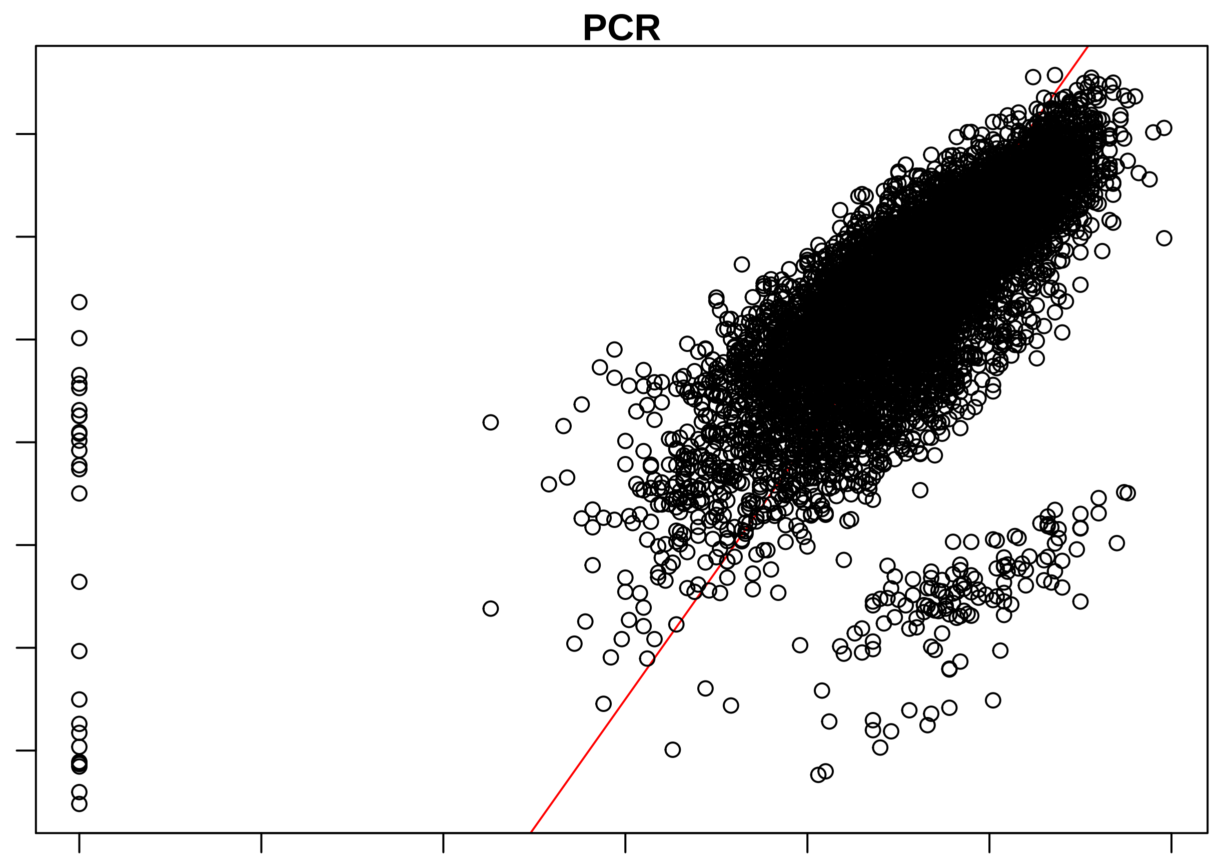
```
knitr::opts_chunk$set(echo = TRUE)
par(mar=c(1,1,1,1))
biplot(pcaT)
```



```r
#Tmax
par(mfrow = c(1,1))
plot(df$Next_Tmax, predict(modTmax),
     xlab = "Observed Next_Tmax", ylab = "Predicted Next_Tmax",
     main = "PCR", abline(a = 0, b = 1, col = "red"))
```



```
#Tmin

par(mfrow = c(1,1))
plot(df$Next_Tmin, predict(modTmin),
     xlab = "Observed Next_Tmin", ylab = "Predicted Next_Tmin",
     main = "PCR", abline(a = 0, b = 1, col = "red"))
```



Analysis:

There are lot of methods in order to compare the four methods.In this I m using errors which gives accuracy as predicting tool among methods.

Results:

An accuracy of model can obtained by getting how much error will get for an model.so, lets find out the errors for each method ,based on the results lets figure it out the best model.

GAM:

```
library(caret)
knitr::opts_chunk$set(echo = TRUE)
#Tmax
mse = mean((testData$Next_Tmax-model1)^2,na.rm=TRUE)
mae = MAE(testData$Next_Tmax, model1)
rmse = RMSE(testData$Next_Tmax, model1)
r2 = R2(testData$Next_Tmax, model1)
error <-array(c(mse,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df1 <-data.frame(type=model,TmaxError=error)
```

```r
#Tmin
mse = mean((testData$Next_Tmin-model2)^2,na.rm=TRUE)
mae = MAE(testData$Next_Tmin, model2)
rmse = RMSE(testData$Next_Tmin, model2)
r2 = R2(testData$Next_Tmin, model2)
error <-array(c(mse,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df2 <-data.frame(type=model,TminError=error)
```

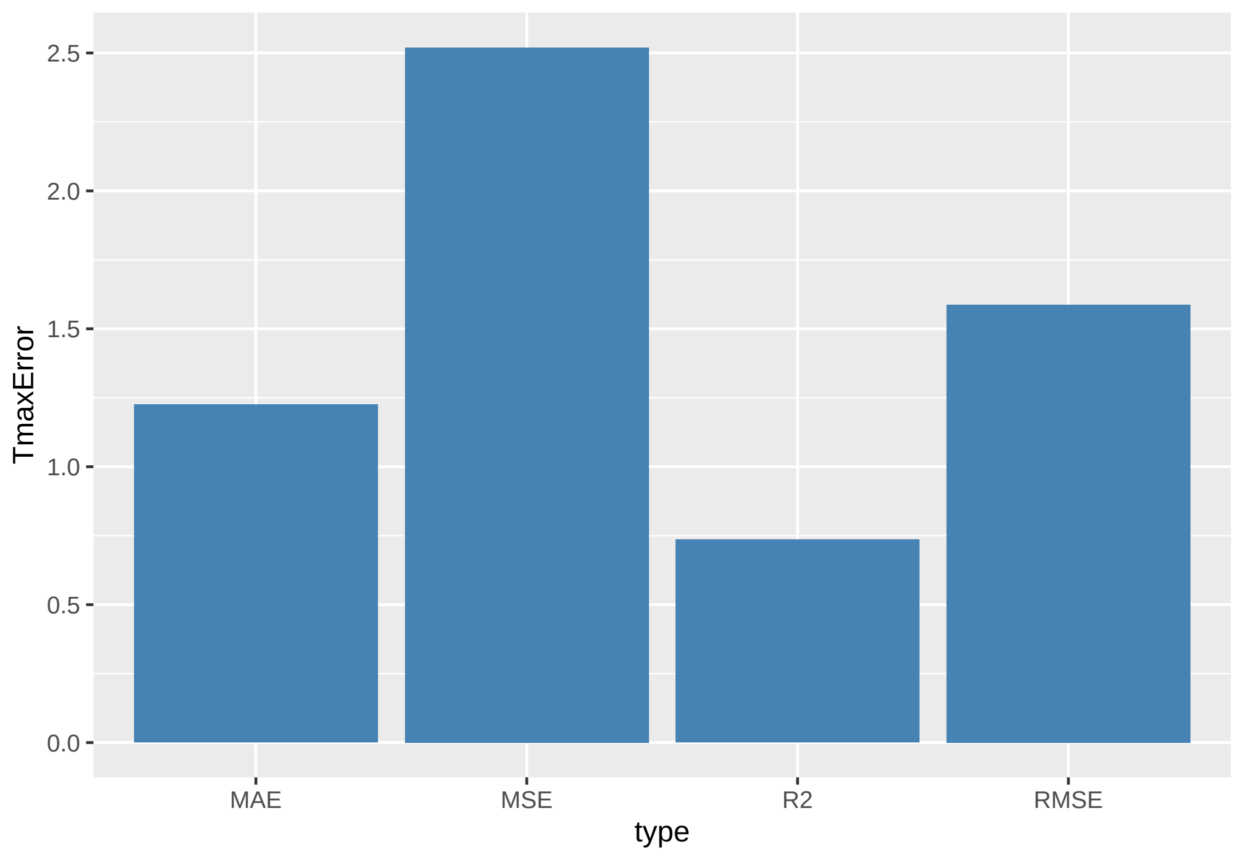
```r
knitr::opts_chunk$set(echo = TRUE)
#Tmax
kable(df1) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TmaxError
MSE	2.5202196
MAE	1.2257280
RMSE	1.5875200
R2	0.7357095

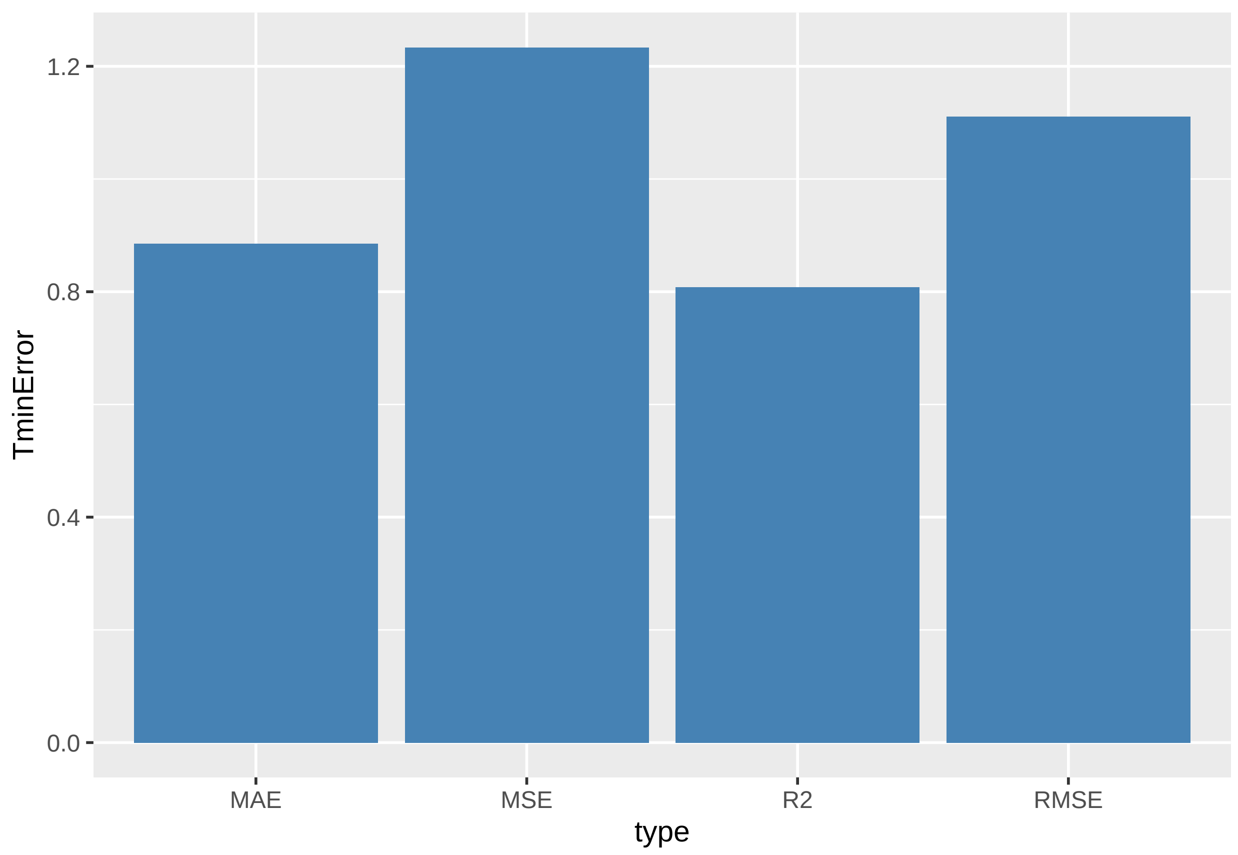
```r
#Tmin
kable(df2) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TminError
MSE	1.2335703
MAE	0.8851082
RMSE	1.1106621
R2	0.8083611

```r
#plots
p<-ggplot(df1,aes(x=type,y=TmaxError))+geom_bar(stat="identity", fill="steelblue")
p
```



```
p<-ggplot(df2,aes(x=type,y=TminError))+geom_bar(stat="identity", fill="steelblue")
p
```



Decision Tree :

```
library(caret)
knitr::opts_chunk$set(echo = TRUE)
#Tmax
mse = mean((testData$Next_Tmax-y1)^2,na.rm=TRUE)
mae = MAE(testData$Next_Tmax, y1)
rmse = RMSE(testData$Next_Tmax, y1)
r2 = R2(testData$Next_Tmax, y1)
error <-array(c(mse,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df3 <-data.frame(type=model,TmaxError=error)

#Tmin
mse = mean((testData$Next_Tmin-y2)^2,na.rm=TRUE)
mae = MAE(testData$Next_Tmin, y2)
rmse = RMSE(testData$Next_Tmin, y2)
r2 = R2(testData$Next_Tmin, y2)
error <-array(c(mse,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df4 <-data.frame(type=model,TminError=error)
```

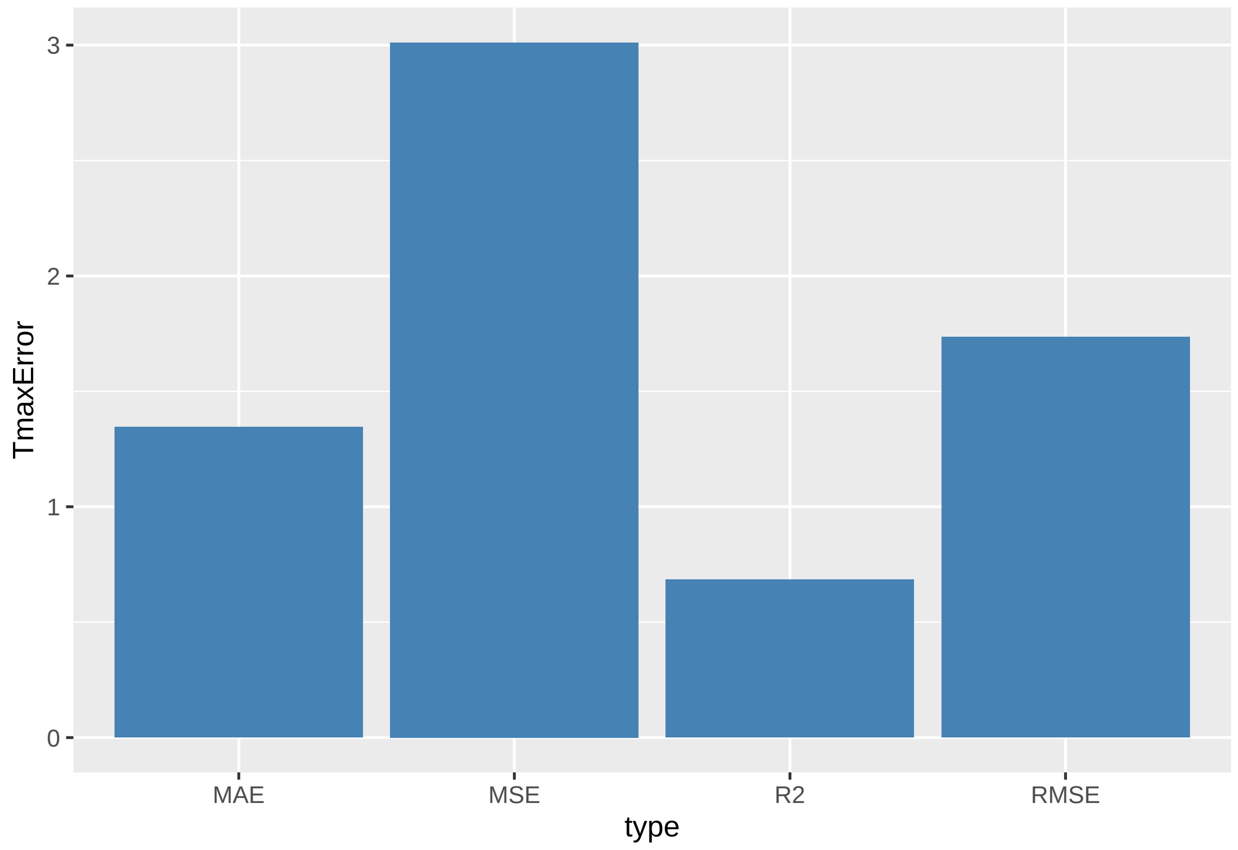
```r
knitr::opts_chunk$set(echo = TRUE)
#Tmax
kable(df3) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TmaxError
MSE	3.0119035
MAE	1.3453767
RMSE	1.7354837
R2	0.6842093

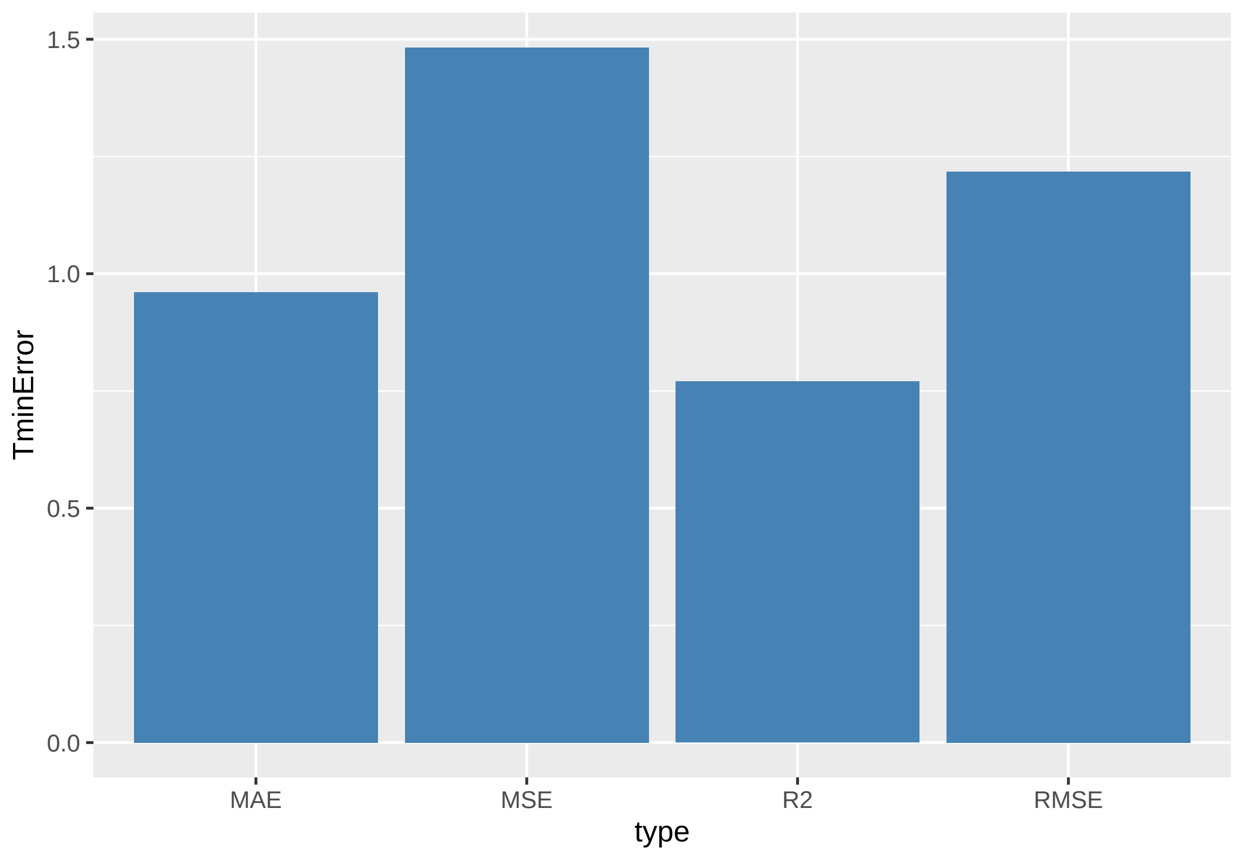
```r
#Tmin
kable(df4) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TminError
MSE	1.4827099
MAE	0.9608989
RMSE	1.2176657
R2	0.7701556

```r
#plots
p<-ggplot(df3,aes(x=type,y=TmaxError))+geom_bar(stat="identity", fill="steelblue")
p
```



```
p<-ggplot(df4,aes(x=type,y=TminError))+geom_bar(stat="identity", fill="steelblue")
p
```



SVM Regression:

```
library(magrittr)
library(dplyr)
library(Metrics)
library(caret)

knitr::opts_chunk$set(echo = TRUE)
#Tmax
mse_Tmax = mean((testData$Next_Tmax-pred)^2)
mae = MAE(testData$Next_Tmax, pred)
rmse = RMSE(testData$Next_Tmax, pred)
r2 = R2(testData$Next_Tmax, pred)
mse_Tmax
```

```
## [1] 1.317732
```

```
error <-array(c(mse_Tmax,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df5 <-data.frame(type=model,TmaxError=error)
#Tmin
mse_Tmin = mean((testData$Next_Tmin-pred1)^2)
mae = MAE(testData$Next_Tmin, pred1)
```

```r
rmse = RMSE(testData$Next_Tmin, pred1)
r2 = R2(testData$Next_Tmin, pred1)
mse_Tmin
```

```
## [1] 0.6543375
```

```r
error <-array(c(mse_Tmin,mae,rmse,r2))
model <- array(c('MSE','MAE','RMSE','R2'))
df6 <-data.frame(type=model,TminError=error)
```

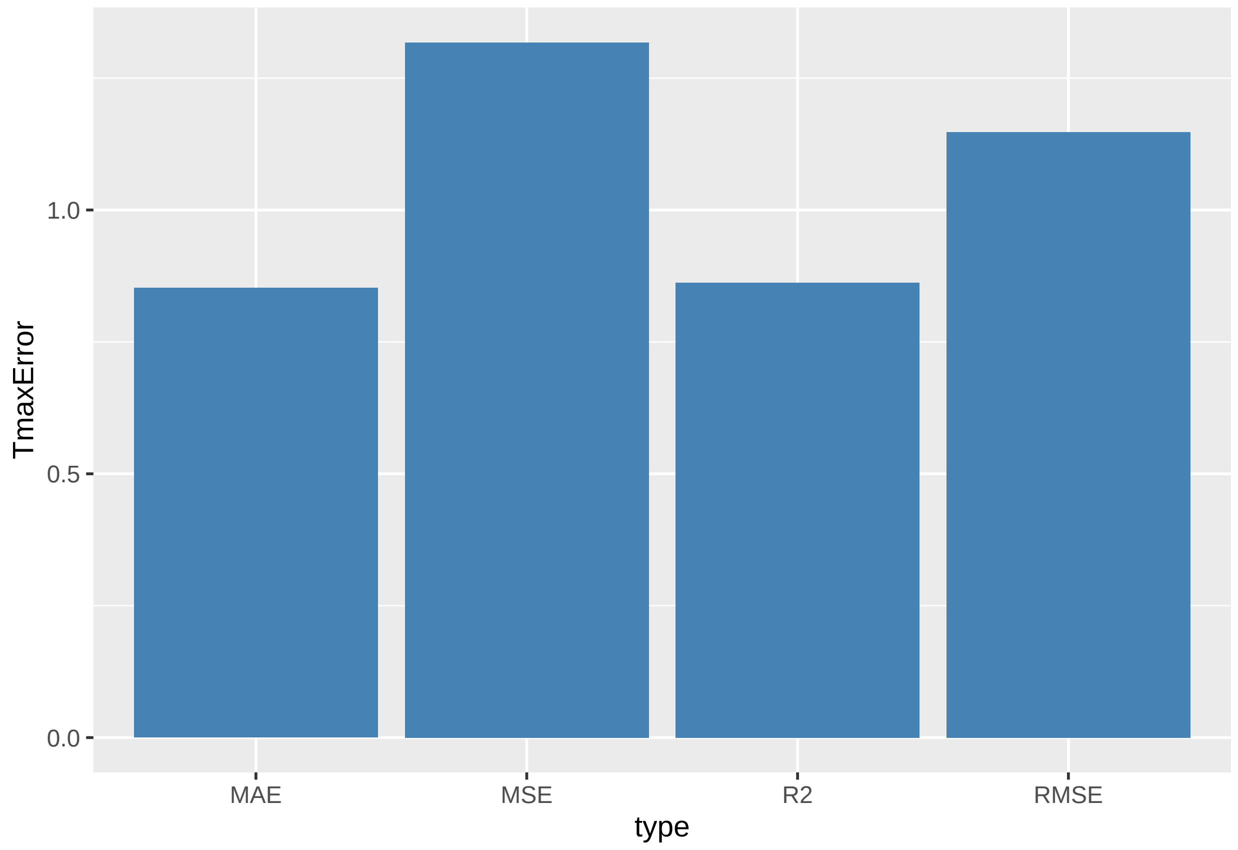
```r
knitr::opts_chunk$set(echo = TRUE)
#Tmax
kable(df5) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TmaxError
MSE	1.3177319
MAE	0.8523370
RMSE	1.1479251
R2	0.8621908

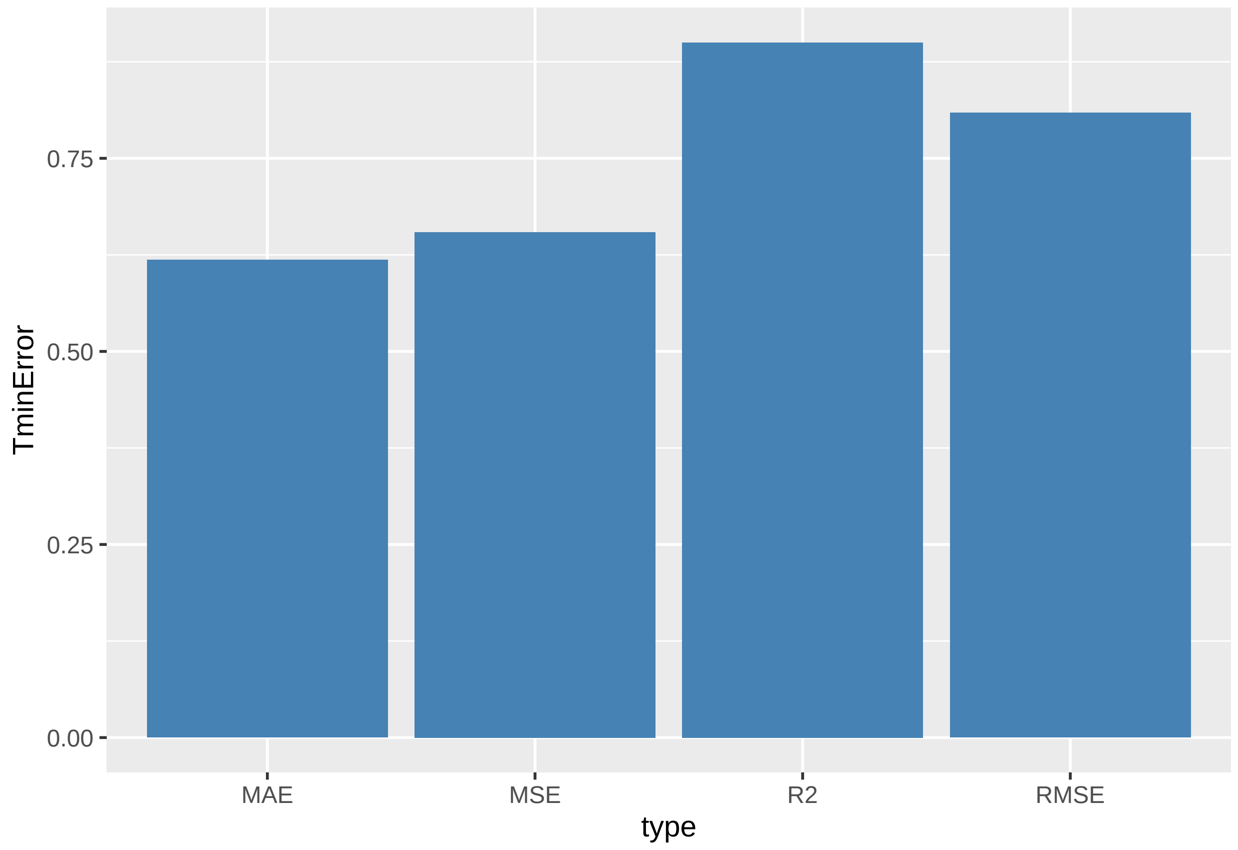
```r
#Tmin
kable(df6) %>%
  kable_styling(full_width = F,"striped")%>%
  column_spec(1,  color = "blue")%>%
  column_spec(2,  color = "red")
```

type	TminError
MSE	0.6543375
MAE	0.6185227
RMSE	0.8089113
R2	0.9001099

```r
#plots
p <-ggplot(df5,aes(x=type,y=TmaxError))+geom_bar(stat="identity", fill="steelblue")
p
```



```
p<-ggplot(df6,aes(x=type,y=TminError))+geom_bar(stat="identity", fill="steelblue")
p
```



PCA:

```
knitr::opts_chunk$set(echo = TRUE)
pcaT$sdev
```

```
##  [1] 2.2572792 1.6763737 1.4718483 1.2429979 1.2008970 1.1582994 1.0546318
##  [8] 0.9781206 0.9590509 0.9216402 0.8561876 0.8508807 0.8066146 0.7756527
## [15] 0.7266193 0.6212984 0.5425142 0.4638440 0.4227333 0.4205104 0.3517972
## [22] 0.3310993 0.1985003
```

```
pr.var=pcaT$sdev ^2
pr.var
```

```
##  [1] 5.09530938 2.81022871 2.16633737 1.54504381 1.44215354 1.34165739
##  [7] 1.11224815 0.95671984 0.91977871 0.84942067 0.73305727 0.72399796
## [13] 0.65062715 0.60163704 0.52797560 0.38601175 0.29432171 0.21515121
## [19] 0.17870346 0.17682895 0.12376124 0.10962673 0.03940236
```

```
pve=pr.var/sum(pr.var)
pve
```

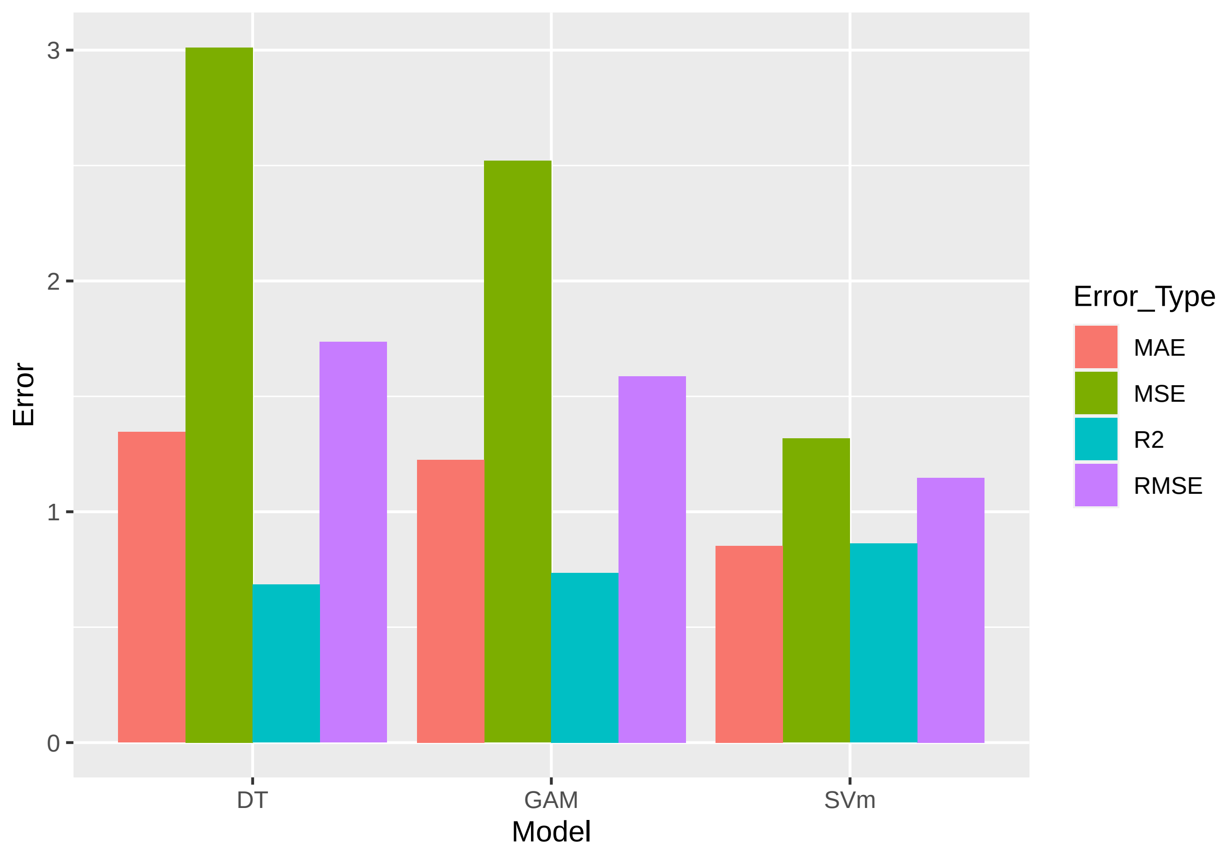
```
##  [1] 0.221535191 0.122183857 0.094188581 0.067175818 0.062702328 0.058332930
##  [7] 0.048358615 0.041596515 0.039990379 0.036931334 0.031872055 0.031478172
## [13] 0.028288137 0.026158132 0.022955461 0.016783120 0.012796596 0.009354401
## [19] 0.007769716 0.007688215 0.005380923 0.004766380 0.001713146
```

Overall comparision:

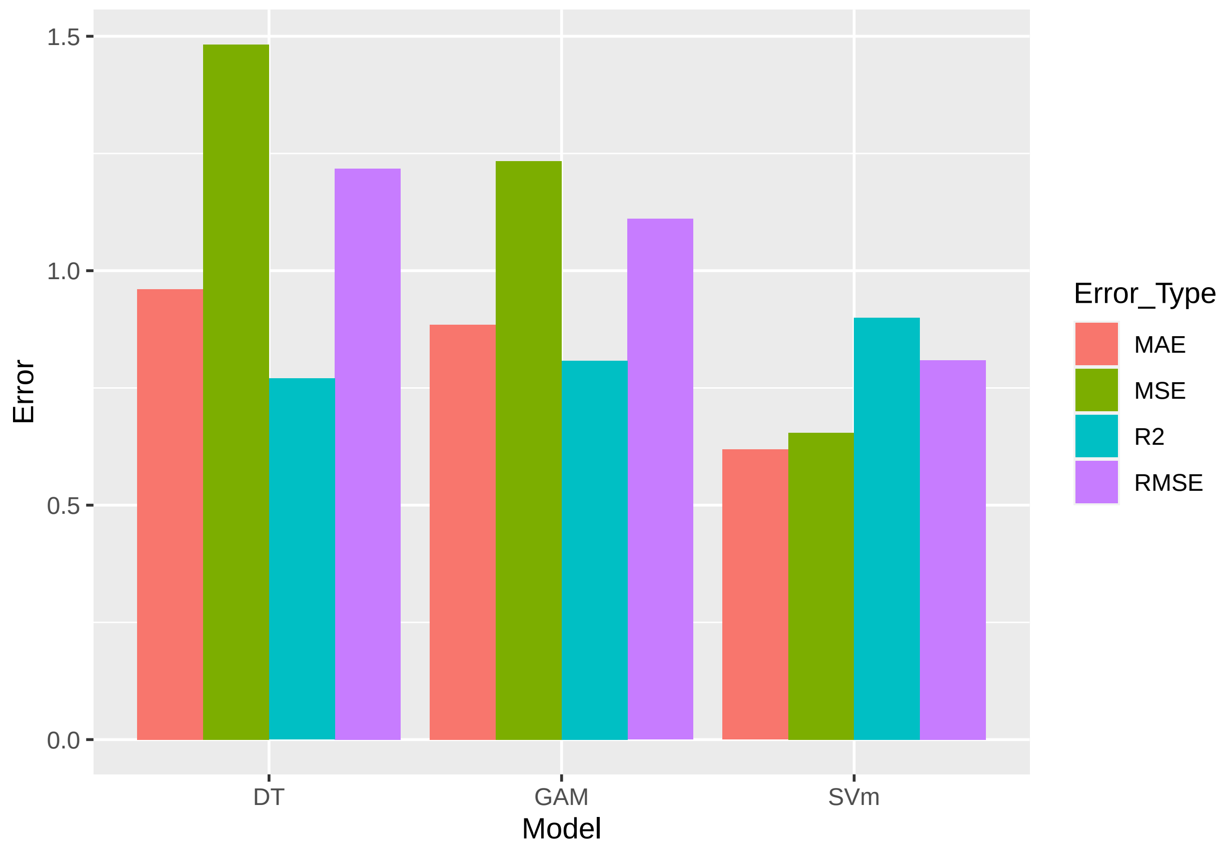
Here , we combining overall error rates for the four methods to have clear view in order to determine for the best model

```
#Tmax
Error=c(rep(df1$TmaxError),rep(df3$TmaxError),rep(df5$TmaxError))
Error
Model=rep(c("GAM","DT","SVm"),each=4)
Error_Type=rep(c("MSE","MAE","RMSE","R2"))
data_Tmax=data.frame(Error,Model,Error_Type)

#Tmin
Error=c(rep(df2$TminError),rep(df4$TminError),rep(df6$TminError))
Error
Model=rep(c("GAM","DT","SVm"),each=4)
Error_Type=rep(c("MSE","MAE","RMSE","R2"))
data_Tmin=data.frame(Error,Model,Error_Type)
```

```
#Tmax
plot<-ggplot(data_Tmax,aes(x=Model,y=Error,fill=Error_Type))+geom_bar(stat="identity", position = "dodg
plot
```



```
#Tmin

plot<-ggplot(data_Tmin,aes(x=Model,y=Error,fill=Error_Type))+geom_bar(stat="identity", position = "dodg
plot
```



Method Analysis:

After Performing four method analysis , I have calculated the error (MSE,MAE,RMSE,R2)rate for each method.With this error rates we can easily find out the accuracy for the methods which gives us the best fitted method for our dataset.Based on analyis the error rate is low for SVM regression model.

Discussion :

Since , the Svm regression model has low error rates with high accuracy among the other methods.so we can conclude that SVM regression model is the best fitted model for our dataset and highly recommended.

I don't think any other methods can better fit for this dataset which we discussed in summer1.Becuase these models can perform easily on nonlinear models also by maintaining a good accuracy level.

References Cited:

Textbook: Introduction of statistical learning with applications in R Cho, D., Yoo, C., Im, J., & Cha, D. (2020). Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas. Earth and Space Science. https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.